An Archive of Scholarly Publishing at the Massachusetts Institute of Technology

by

Lydia Sandon

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering
In Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author .....
Department of
Electrical Engineering and Computer Science
May 1999

Certified by .....
Harold Abelson
Professor, Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

An Archive of Scholarly Publishing
at the Massachusetts Institute of Technology

by
Lydia Sandon

Submitted to the Department of
Electrical Engineering and Computer Science
on May 1999, in partial fulfillment of the
requirements for the degree of
Master of Engineering
In Electrical Engineering and Computer Science

Abstract

This paper details the research into, design of, and implementation of a computer system built to archive scholarly publishing completed at the Massachusetts Institute of Technology and within other institutions of higher learning.

The archive will provide a central, stable repository for researchers looking for permanent housing for their published work. The Web interface to the system provides easy means of collaboration among researchers within a single institution through simple searching on relevant categories. The database will also provide an easy mechanism for creation and maintenance of an academic *curriculum vitae* for its users as well as a forum for relevant feedback on the content.

Fundamentally, however, the system has another purpose: to increase the libraries' role as knowledge-keepers over commercial journals which would like to maintain that position at a cost to both researchers and libraries. This system serves both to attempt to change a *status quo* which is rather undesirable, especially in the face of recent legislative decisions, and to show that technological systems are viable mechanisms for change of social establishment.

Thesis Supervisor: Harold Abelson
Title: Professor, Department of Electrical Engineering and Computer Science

Acknowledgments

I would like first to thank Hal Abelson, Class of 1922 Professor of Electrical Engineering and Computer Science. As he has done for so many other people, he has helped me tremendously and taught me much. I couldn't have asked for a better advisor and am lucky to have had the opportunity to work with him.

I owe thanks also, for their feedback and guidance, to Ms. Anne Wolpert, Director, and Mr. Eric Celeste, Assistant Director for Technology Planning and Administration, of the MIT Libraries. Without them, this thesis could be neither complete nor useful.

To my friends at the Massachusetts Institute of Technology and elsewhere, I owe a special thanks for their invaluable contributions "to the sanity of the author" [20]. This thanks goes particularly to Benjamin Adida, for his practically endless supply of help and care.

This thesis is dedicated, with infinite appreciation and love, to my sister Mara Sandon, my parents, Mioara and Zalman Sandon, my maternal grandparents, Dorica and Maurice Bratu, my paternal grandparents, Bertina and Izu Sandon, and the rest of our extended family.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Commercial journals have long been the publishing medium used by academic researchers as a forum for scholarly debate. In order to have one's academic work critiqued by others in the field, it is frequently even necessary to publish the research in a journal which requires a paid subscription. People of similar academic interests can communicate in a public forum through these journals, independent of separation in distance and time.

Libraries are sometimes little more than "brokers of knowledge" [2]. They facilitate the use of commercial journals by academic researchers and other members of university communities who are looking for additional information. Rather than taking an active role in the research completed within their universities, libraries are frequently bystanders, purchasing journals which print articles written by their faculty for use by their faculty.

Should this still be the case with the recent popularity of a basically free, collaborative publication source such as the World Wide Web? Digital media are increasingly popular mechanisms for publishing. Their low cost, resilience, popularity, and accessibility make them appealing alternatives to expensive, low-circulation academic journals. Some kinds of digital media may actually allow university libraries to provide a unique home for the research completed within their hallowed halls. Indeed, not only would this allow libraries to be more actively involved in the production of interesting research, but it may also lend control of the intellectual property back to its source rather than to the commercial journal. But how should libraries get more involved in university research?

1.2 The idea

> The Internet revolution is injecting more competition into publishing and giving power back to scientists and learned societies. It presents new challenges to the guardians of the archives and yet could spell the end for many print titles. [2]

As Declan Butler explains above in *The Writing is on the Web for Journals in Print* [2], technology can play a pivotal position for libraries right now. This thesis presents one possible mechanism to help libraries increase their involvement: an on-line archive of research articles.

The online archive can provide numerous features which make it desirable to researchers and

publishers alike. Permanent pointers to articles helps to negate the problem of perceived immaturity of the Web; it will also provide a convenient mechanism for researchers to put together *curriculum vitae* and electronic journals on the Web. These can be used by researchers to allow commentary to be contributed outside of the realm of commercial journals.

This archive of scholarly publishing may also help to alleviate some of the problems currently coming about because of the shake-up caused by the increased use of the Internet. While printed journals are expensive and generally have low circulation, Web archives may provide a one-stop-shopping sort of convenience which makes libraries a more important player.

Libraries have long described a desire to keep research in the hands of well-meaning, non-profit communities, and could use such a system to help. In addition, this system may reduce the overhead of the library, providing nearly as much information with far fewer expensive journal subscriptions. Finally, combining these two issues, the libraries may actually help researchers avoid losing control over their intellectual property through changing copyright legislation [14].

1.3 Overview: Scenario

The system operates through a series of interactions among two types of people: there are users who have privileges only to make requests, and there are those who can make administrative decisions. Imagine an interaction among three characters: Alice, Bob, and Charles. Alice is a "regular user" of the system; she would like to upload an article into the archive and have it designated as a University of Massachusetts Department of Computer Science Technical Report. She logs onto the site and gives her appropriate identification information; Alice is then taken to the central homepage shown in Figure 1-1.

When Alice goes to the "upload article" page, she notices that her HTML format is available. She enters the article's name and format, and her document is uploaded. The system presents her with a page on which she can enter the document's metadata, as shown in Figure 1-2.

The system verifies the metadata and she is given her permanent URLs, so that she can link her work permanently to the archive from anywhere else on the Web. Currently, the system generates those permanent URLs simply by creating a unique identifier for each article and pointing the user to that identifier. The assignment of Alice's permanent URLs is shown in Figure 1-3.

Because she is interested in having her document considered as an official" publication," she indicates to the system that she wants the article to be considered for "University of Massachusetts Department of Computer Science Technical Report." She is told that her request was received, and that two members of that realm's administration would need to approve it.

Finally, she asks that she be considered for the Department of Computer Science's "Artificial Intelligence Group" realm, because she has recently been made an Associate Professor, and she feels that she should have administrative privileges to approve articles and new members into this realm. This request is received as well, and this time the system informs her that one person from the "AI" realm will need to approve this request.

Bob and Charles, two members of the Department of Computer Science realms, both receive notification through email that they have requests pending for them. Since they are the only two

Figure 1-1: Screenshot of homepage

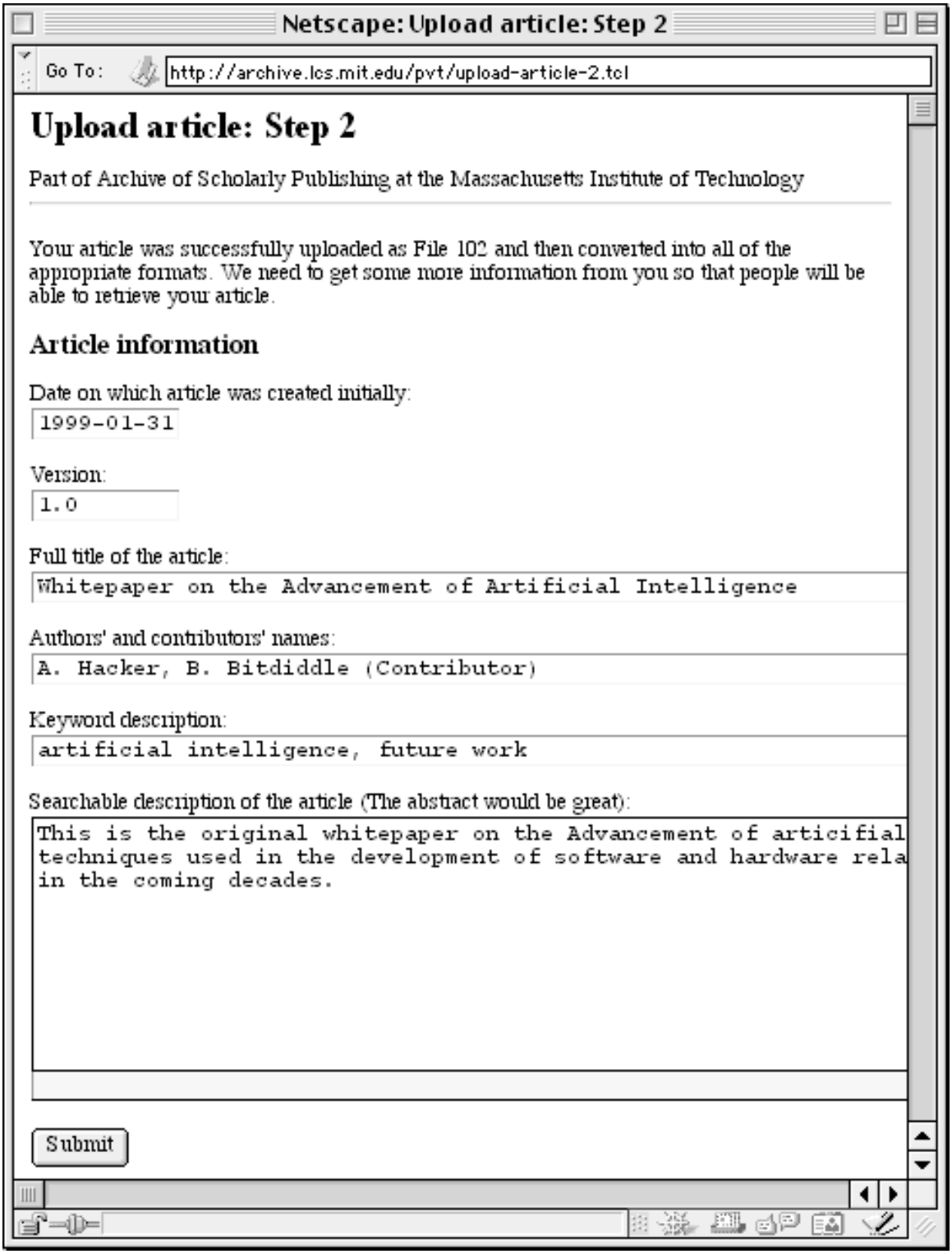

Figure 1-2: Screenshot of metadata verification

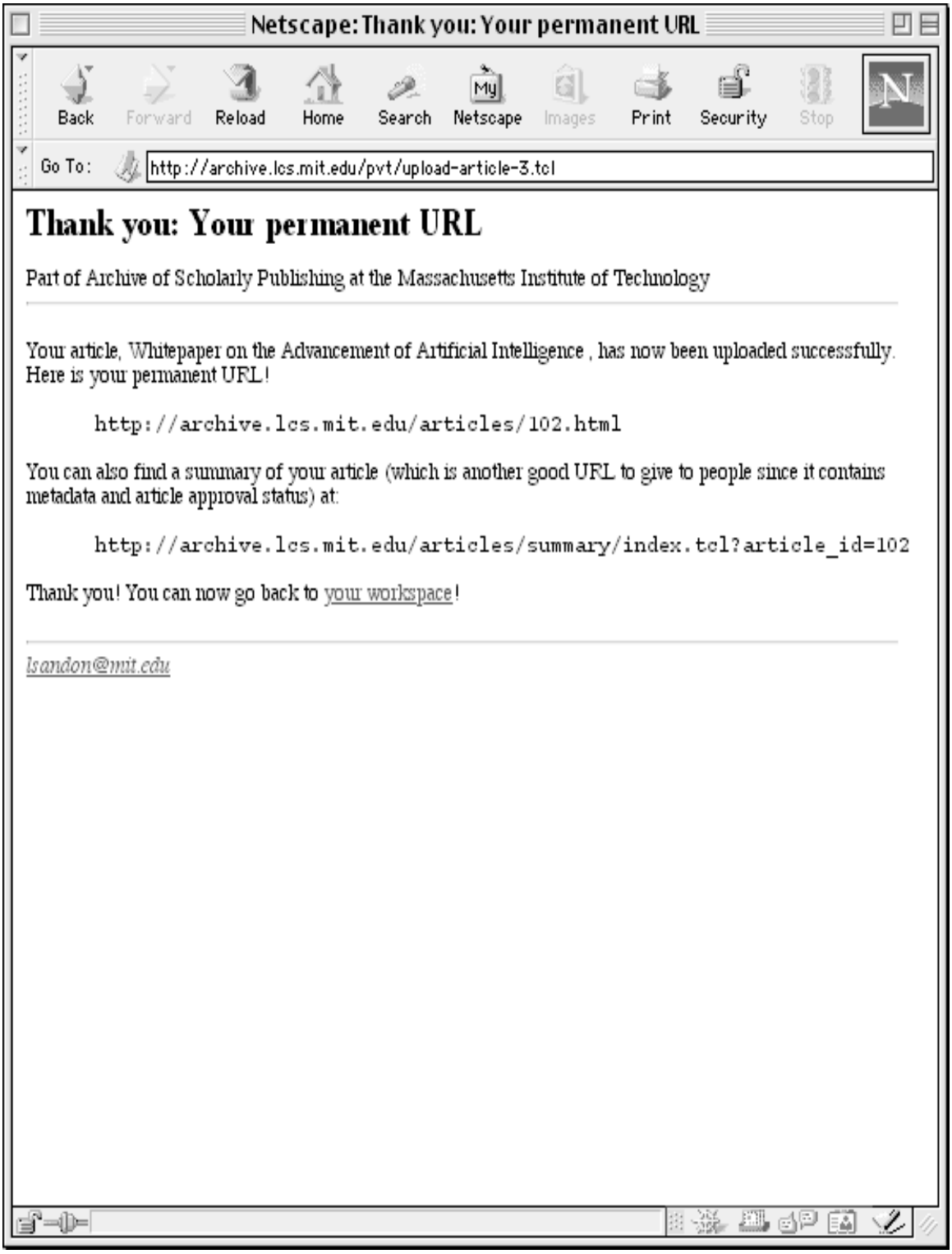

Figure 1-3: Screenshot of permanent URL distribution

members of the realm, they will both need to approve the article type status in order for it to be approved. Bob approves the article and enters his comments about his decision into the system. Charles feel that it is a good article but needs more work before it can be officially approved by the department; he notes this in the system before denying Alice's request. Alice receives this overall denial notification, with the reasons why. She is disappointed but since she has the comments available, she knows what to do in order to resubmit the article later for approval.

Since Charles is also a member of the AI group realm, he approves Alice's request to be an administrator within that ream. She receives notification again, this time with the news that she has been approved. Now, any AI-wide requests will also be sent to her. This means that her approval will be considered sufficient, as before, to approve another person into the realm, and so on.

Since she is a new professor, Alice is creating a new partnership with another professor in the same group within the Artificial Intelligence group, called the "Advanced Artificial Intelligence Group." Since she is a professor with administrative privileges into the AI group, she has the right to create this partnership. Alice does this at the "request a realm" page. Because of her privileges, she can also go to the "manage realms" page and approve her own request!

It is through use like this that the real-world analogues of article type approval can be streamlined, that the libraries can keep track of what is being published, that researchers can have a convenient and permanent storage mechanism for their work, and that – later – commentary on scholarly work can be done online.

1.4 The paper

This document outlines the design and implementation of a prototype research article archive. The rest of this document presents the following information:

Chapter 2: Goals and design This chapter will detail the high-level goals of the system. It will then describe in detail the system's design.

Chapter 3: Implementation This chapter will present the technological choices for implementation of the archive, comparing along the way with the requirements presented in the goals and design section.

Chapter 4: Potential effects of the system After the discussion of the precise features and implementation of the system, this section will put forth the possible effects of this archive, and the reasons why these effects may happen.

Chapter 5: Conclusions The paper ends with the presentation of the conclusions and future work outlined throughout the paper.

Chapter 2

Goals and design

> Libraries and scientists are now striking back, the ultimate goal being to return control of scholarly publishing to non-profit societies and what they consider to be responsible publishers. [2]

As Butler describes here, libraries wish to have increased influence over the scholarly publishing world. However, this influence comes at a price, and may be difficult to obtain. New copyright laws have made the situation even more potentially divisive. Research libraries may encounter difficulty, as times wear on, trying to pull control back from a historically journal-controlled environment.

What might cause difficulties in the adoption of such a system? First, researchers may not use any other publishing mechanism; such a tool may lay barren and empty, providing no use to anyone. Since it is not the researchers themselves who are paying for the journal subscriptions, there may be little obvious reason to switch; the real advantage is to the libraries. The system must, unfortunately, be relatively widely used in order to gain mindshare and popularity, hence retaining control from commercial academic journals.

2.1 Goals

If an archive tool, as outlined above, is going to make a difference for the library world, great pains must be taken in three forms:

- Any alternative tool must offer features which provide improvement to the researcher's academic life. It is not enough to provide a service similar to the ones provided by paper academic journals; it must be better in order to overcome an existing disadvantage.
- The tool must not add much overhead to the researcher's life. It must be easy and fast to use. Regardless of how useful the archive's features or advantages, it will not be used if its use is cumbersome.
- The archive must aim to retrieve control of intellectual property from the printed commercial journals back to the authors or even the non-profit libraries.

Finally, most libraries scarcely have the resources to begin administering an expensive or controversial service. While journals are expensive, unless an online disintermediation tool is able to replace it entirely, a savings is unlikely to occur immediately; the cost of any technological solution must be minimal. Put together, the above goals must be used to guide the design of the system.

2.2 Design

Most of the actions provided by the archive system come in the form of a dialog of interactions between a user and a librarian-administrator, or parties acting on the administrator's behalf as outlined in the initial usage scenario. In the case of the user, he or she is usually requesting that an action be taken. For example, the user may request that an article be uploaded or that he or she be given the privilege to operate the system in an administrative manner. The librarian-administrator, or faculty members acting on the administrator's behalf, will then check that the article bears the qualifications necessary to be admitted into the archive or that the user is approved to be given greater privileges into the system.

In this section, we will look into the chronology of these interactions in greater detail, from the side of the user. Some of this outlined design will be implemented and the implementation discussed in Chapter 3. Other parts will be left to future work, and will be discussed in Section 5.1.

Library systems must take into account the usability of the system so that the libraries encourage many people take part in its use. The thesis.mit.edu site, for example, is an archive site that rarely used even though it is financially beneficial to its users, because it is somewhat difficult to use. Second, the site must be cost efficient in order to allow the site administrators to operate on a library budget, and cannot be made obsolete by another free service later on. Third, the system must protect the privacy of its users, not allowing any more information about readers or publishers of articles to be viewed than the reader or publisher desires. Clearly, as well, the system must also be robust and scalable, in case it should be used outside of the Institute community. Even inside the Institute, there are roughly 10,000 articles that are already candidates for uploading into the system.

2.2.1 Users, user identification, and permissions

The service's use begins with its end users. These are the people who are in need of a permanent storage and retrieval mechanism for their articles. Generally, this refers to faculty, research staff, and graduate students anywhere who are using the service for permanent storage of their research output. However, this will also be used by those people who are following links to the site from the *curriculum vitae* or other HTML pages of those researchers mentioned above.

It will not be necessary at this time for passers-by to identify themselves to the system in order to read an article. Most of the articles originally uploaded into the archive will be production-quality, finished works that are ready for viewing by the general public. In this case, there is no reason for readers to identify themselves before reading the document.

Two high-level changes to the system might require readers to log in. The first reason is if draft versions of the articles were stored on the system. Here, researchers might want to use the site as a convenient way to collaborate with scholars in their field. However, the article might not be ready for publication and, therefore, not ready for public viewing. In this first case, the archive could require the use of some sort of identification system to match a user to the articles that he or she had the permission to view or edit.

It will, however, be necessary for users of the system to identify themselves in order to upload

information into the system. Identity is important in determining whether a user is allowed to complete a desired function. For example, some users are authorized to approve articles for "technical report" status, while others are not. Still others are approved to grant additional administrative privileges to particular users. At the same time, identity is important for record-keeping and notification to the user about the status of a particular transaction within the system. Thus, at some point in the beginning of the user of the system by a new researcher will need to identify himself or herself.

In addition to identifying users and their relationships to the articles that they upload as well as the privileges tha they have, users must also establish relationships to certain other people. For example, users may very well have assistants who have the right to act on the their behalf. This relationship must be established at some point through the use of *groups* to which a user can add others and establish privileges for.

2.2.2 Article uploading and article type definition

In addition to users, we have the all-important articles within the archive. When a user logs on, it is likely the the majority of his or her requirements with the system will be in the uploading of articles into the system in exchange for a permanent URL which will point to the centrally-stored article.

When the user uploads an article, it must be identified by the author as being of a particular type. Examples of these types could be "Laboratory for Computer Science Technical Report" or "public work in progress." In the case of a public work in progress, there is no reason to believe that the article needs to be approved by any sort of committee or board, and bears no representation of the institution or organization where the author may be employed. On the other hand, for a technical report to be considered approved, for example, three faculty persons within the department may be required to approve it. In this case, the approval mechanism is vastly different and thus different actions need to be taken.

In addition to the article type, there is the format of the document to take into consideration. If the system will be able to read and convert the format of the document for any reason, it must be aware of what the format is and have the software necessary to do any conversion. While it may be possible to identify the format of the document in several ways, the easiest by far is to ask that the document be uploaded with a file extension particular to its format (such as .ps for PostScript, etc). The archive currently does no verification of format, however.

2.2.3 Article editing

Once a user uploads an article, will he or she be able to edit its contents? In theory, there are several reasons why users might want to be able to edit their work. First, he or she may make changes to the article and wish to keep the same URL for the article regardless of the version that it is at. Second, the user may want a living mechanism for collaboration among multiple writers of the same document; in effect, wanting to use the system as a shared resource for revision control.

The ability to edit an article would be useful to certain users because of these aforementioned reasons. Editing ability would provide additional, appealing flexibility to the system. In addition, it would indeed make the system usable as a shared community of writers collaborating on works of

research. Accordingly, user permissions would have to be changed in order to allow only approved viewers of the not-ready-for-prime-time research articles.

For the prototype version of the system designed in conjunction with this paper, the ability to edit articles will not be included. There are several reasons for this. Importantly, article editing is not critical to the proof of concept that the paper hopes to further, and may potentially increase the complexity of its use. However, the ability to make changes and view the audit trail of changes made to the system will certainly make the system more usable, and is a desirable function for later implementation.

2.2.4 Metadata collection

Besides physically uploading the article, the archive must be made aware of some metadata about the article, such as its title and authors. This information is what makes the archive a useful one for the libraries. It allows the article not only to be stored in this archive service, but also to be recognized by similar services, such as the libraries' Web site. It means that the data will not have to be entered by a librarian-administrator, and also provides meaningful statistical information about the use of the service.

There are three ways to do this metadata uploading. The first is by intelligently guessing from the content of the article what its "metadata" are. For example, if in a latex document you could parse out a latex command called title, you might imagine that its contents contained the title. Clearly an intelligent parsing tool could pull this sort of data out of the document itself. However, there are other document formats in which even this obvious field would be hard to define. As well, there are other fields, such as the keywords used to describe the document, which would always be hard to garner. To do this, the document would need to be analyzed as follows: which words are most common to this document, removing those words which are inherently more common to the language in which it is written? Those, then, are the probable keywords for the article. This mechanism is ideal in terms of least burden required on the user uploading the article; however, it is technically difficult to implement and likely to be fairly inaccurate.

The second way to upload metadata is to ask the researcher or author to input particular metadata by hand. He or she would be presented with a screen full of brief questions about the article which would then be used directly by the system or indirectly, by a librarian-administrator or other approved user, to catalog the article and approve it if necessary. Clearly, this mechanism places a much greater burden on the user uploading the article, but it is much easier to implement and is likely to be nearly perfectly accurate.

Finally, the metadata can be uploaded through the use of predefined metadata tags which are included in the document itself. For example, in the case of HyperText (HTML) documents, this would take the form of META tags in the head of the document. This could also be in the form, later, of XML tags appropriate to this cause. Clearly this would be the easiest way to go about this task, because it would be automated entirely by a client uploading the document, thus making it a distributed task. It would require little to no effort on the part of the user uploading the document as well as the server receiving the document, but would be more accurate on the whole than the guessing mechanism described above.

2.2.5 Approval functionality and notification

Once a user has identified himself and has tried to upload an article, identifying the article types with which he or she would like the article identified, the article is entered into the system with an approval status of "pending." Depending on the approval mechanism attached to that article type, the realm members whose approval was necessary would be notified. The necessary administrators would then return to the service in order to review the appropriateness article. When the user had reached a decision, the system would notify the author(s) about the decision and a permanent location pointer (see below) would be assigned to the article.

It is likely that not only will each article type require its own approval protocol, but also that these protocols may need to be changed over time. Given that the system should be extensible in order to prevent obsolescence, these mechanisms should also be flexible; they should be mutable at the will of the librarian-administrator who is running the system. Further, a user should be able to suggest a new mechanism through regular use of the site.

2.2.6 Permanent location pointers

One of the key features of this service is that it will provide a permanent locator to every article uploaded into the system. This will provide users with a means in which to link pointers to their work directly from their pages.

There are basically infinite ways to provide a permanent URL pointing to an article. Fundamentally, the manner in which this is done needs to have only two characteristics: they must be unique URLs, and they must be persistent. Additionally, it seems that it would be ideal if they URLs would be human readable (and easy to remember) because this would make them that much more usable. Finally, if they can be somehow logical, dividing parts of URLs into parts of the collection of documents, that would also be ideal. Different choices include combinations of submission or creation date, author name, and a unique identifier.

Although each URL handed out by the system must be persistent, there is nothing to say that the protocol used to hand out the URLs needs to be. Therefore, the system's design must allow an administrator to hand-change the system. In addition, the software used to resolve the URLs into system-specific article pointers must be aware of the possibility for change and handle it appropriately.

2.2.7 Additional formats and conversions

What is to be done when formats start forming loops so that they can be converted only to a few other formats but not to every one? Unfortunately, short of writing a converter, little can be done. It must be accepted that the system will convert as much as possible to allow for maximum flexibility, but that almost nothing can be done to prevent loops.

In order to maintain the system's extensibility throughout its use, users and administrators must be able to suggest and approve many kinds of changes to the system. The crucial point here is that these changes can then be applied directly without any programmer modification to the system. This is as true of the document formats accepted into the system as with any others part.

Thus, the user must be able to request that a document format be added to the system. Indeed,

he or she must also be asked to provide server commands which the administrator can approve to be used to convert this new format to and from existing formats. In this way, the system can be made to accept arbitrary formats without additional modification. Also, indirectly, these new formats can provide links between two format-loops!

2.2.8 Additional user privileges

What are user privileges? First, they are the objective titles provided by the system, such as "faculty person" or "librarian administrator." Second, a user privilege is the mapping of one of these titles to a particular user, such as "Harold Abelson is a Faculty person" or "Ann Wolpert is a Librarian administrator."

There are certain user privileges, in the former sense, built into the system. These should be augmented through the system; an ordinary user or administrator should be able to request an "undergraduate researcher" title or something similar, which would allow a particular kind of privilege to be conferred onto a user with such a title.

There are also certain user-privilege maps which are built into the system. These are the boot-strapping users which allow the system to bring other people into the fold. However, the administrators and people of privilege must be allowed to modify appropriate privileges throughout the system; again, this must be done without the benefit of programmer intervention to maintain flexibility. In addition, it would be ideal if an average-Joe user could request additional privileges and be granted them through the ordinary use of the system.

2.2.9 Metadata exporting and integration with other services

When articles are provided to users who are viewing or linking to them, they should be exported with this metadata attached. This way, the service is adding value to the quality of data that it provides. How this is done depends in each case on the format in which the data is presented. Clearly, it is easier in some formats to add metadata than in others. For example, it is easy to add meta tags to the head of an HTML document than it is to do so in other formats. Thus, these must be dealt with accordingly within the framework of the standard chosen to represent the data in the service.

Finally, once all of this information is stored and cataloged, it must be available for certain kinds of integration into existing library systems. This may mean that the service provides a live set of hooks into the system which the libraries use as an augmented data set. It may also mean periodic dumping of information into a more central repository. Clearly, providing a set of live hooks into the system would provide the most accurate and up-to-date information available to the archive. However, it would also put an additional strain on the service, when its primary purpose is to be accepting new documents into its storage. The latter choice, periodic dumping or export, is easier for the service itself to handle since the export can be done at periods of low use. However, it provides less up-to-date information to the wider community. As with all design choices, there are tradeoffs which must be considered relative to the implementation of the service as a whole.

Chapter 3

Implementation

This chapter will include an overview of the implementation of a prototype of the archive service. It will include a introduction to the technologies and standards which were chosen, an discussion of the data model chosen for the database part of the implementation, and issues related to future integration with other services.

3.1 Interesting technology and implementation choices

The service will be provided primarily for use on the World Wide Web. Why? Because the popularity provided by the current use of the Web allows an online service to counteract the high-cost, low-interest publication problem encountered by commercial journals. Because the Web is a great interface to a collaborative publication tool. Finally, because a Web service can be easily interfaced with other Web tools, and there certainly are a lot of up-and-coming Web library services.

What other options did we have? We could have had any kind of client-server architecture, running on computer(s) owned by the Libraries. We could have used a Java applet or other program to access the information locally or through a network. Why not use those options? Because they are slower and less capable of tolerating the high load that the service may attract. Because they require the downloading of software specific to the service, which an HTML document does not.

What are the dangers of using the Web? The Web is unreliable, immature; those are the standard arguments. It is poorly typed, providing little information outside of scores and scores of strings. Can we work around these problems? We hope so. Chapter 2.2 points out how this can be done at a high level; the following sections describe how this can be done at a practical one.

3.1.1 Information storage

The archive must be able to store and access a tremendous amount of information about the articles and users involved in the service. Of course, there are many options for this kind of storage. A common answer for the Web is text files stored in a normal file system and accessed through Perl-type Common Gateway Interface (CGI) programs. Another common answer is a relational or other type of database program which is accessed through any number of means. Still another is through persistent objects stored in serialized form in the file system and accessed through a related programming language; Java's object serialization is the perfect example of such a system.

This system relies upon a relational database for information storage; why? Because, of all of the options outlined here, it is the solution most closely related to the needs of the archive. It is an immensely robust yet fast at data access and easy to program using the techniques and tools outlined below. Specifically, the relational database chosen for this prototype is the Oracle 8 version. It is particularly robust and easy to program. It supports transactions which allow for more complex uses of the site. It uses ANSI SQL, which is a standard technology which may allow for integration at a lower level with other database-backed services. Oracle is a particularly cost-efficient solution within MIT because of the site license that MIT holds; this is a particularly important facet of a service provided by a non-profit organization. Finally, Oracle's future focus on Internet-related tools may allow interesting extensions to the service through direct "business-logic" programming.

It is perhaps unnecessary to say that the system will run Oracle on Unix. This is partly due to MIT's previous use of Unix as a standard. It is partly because it is one of the most stable operating systems currently available, an important feature of the operating system running a "highly available" Web service.

3.1.2 Web server

What software should the system rely on to serve this stored data to the Web? AOLserver [17] was chosen, largely due to successful past experience.

There are good reasons for this success. First, AOLserver is based on threads rather than processes, which makes it more efficiently multithreaded. That is, it is desirable as a Web server which may get many simultaneous requests of all kinds. Second, it maintains open connections to the database in a pool; it hands database handles out when required, rather than opening new connections for each request. This is a much more efficient solution. When the two, efficient multithreading and database-connection pooling, are combined, the Web server is very efficient at doing its job.

In addition, there are development concerns as well. AOLserver provides a Tcl interface for the developer. First, Tcl is a fast development tool because it is an interpreted language, not requiring compilation for changes. Second, Tcl is very easy to learn, not requiring the memorization or understanding of very much syntax, making it a very good prototyping tool. Finally, Tcl is weakly typed, dealing with most things as strings; while this may sound undesirable, it is actually perfect for the Web, where all information is passed around in string form.

Finally, there are ancillary reasons to use AOLserver. Its next release, and subsequent releases, will be open sourced, providing additional programmer effort and attention. Open-sourcing has historically improved the quality of the product (such as in the case of Linux) , so it can be reasonably assumed that it will improve even further the quality of the server. Finally, the next version is expected to support digital certificates, a great way of dealing with security (discussed further in Section 5.1).

3.1.3 Standard information selection

What metadata does the system need to collect about each article? This is an important question as the utility of the article metadata is strictly related to the utility of the entire archive. Provide

not enough data and cataloging will be difficult or impossible. Ask for too much data and the user will be burdened in an undesirable way.

The system uploads information based on the Dublin Core Metadata Institute [8]. It was chosen precisely because of its birth in the same realm: it was meant as a lightweight metadata set which would provide the bare minimum cataloging information necessary. Thus, the system relies on the Dublin Core and processes data according to it.

This is done currently through the manual entry of data according to the DC element set. The parsed information is related to the article in the database and provided for cataloging and searching purposes. However, the data could be parsed out of headers or imported in another way; those are left to future work.

3.1.4 Permanent information access

One of the key features, as outlined by the Design chapter, is the designation of permanent URLs by the system. In order to do that, the system relies on a scheme similar to that provided by the PURL [4] software mechanism.

The PURL software was not directly used by the archive system. This is because the two primary benefits of the system were its URL parsing software and its indirection through `purl.org`, neither of which were directly used.

Instead, the system pulls inspiration from the PURL mechanism by parsing URLs in a similar vein, and by creating its own permanent URL domain. While this service uses the base URL `http://archive.lcs.mit.edu/articles`, there is no reason why this couldn't be mapped to `http://purl.mit.edu` or the like. Similarly, the service could directly use the PURL parsing tool.

3.1.5 Community building

Reliance on the ArsDigita Community System [13] was chosen because of the functionality it provides in allowing users to better deal with each other and with the archive system. Rather than starting from scratch, several of the system's components can be used to build a community of users at different levels of administrative control. (In addition, the ACS relies on most of the same core technologies as listed above, providing reduced design and prototyping time.) Specifically, the user groups module of the ACS, which allows users to be placed into groups and those groups categorized into types, dramatically reduced the amount of time required to build the administrator privileges that are designed in Chapter 2.2 and implemented in Section 3.2.2.

3.2 Conceptual modules and the corresponding data model

There are several pieces of the puzzle in the implementation of this system according to the specifications in Chapter 2.2. There are *users*, people who are involved in the use of the system at all levels. Those users have *roles* within *realms*, the parts that they play and the organizations within which they play those parts. They can make *requests* or, in some cases, *approve* the requests that others have made.

These modules, and the implementations of their corresponding data models, are described here in detail. Each section is described on its own in relation to the chronology set out in Chapter 2.2. Generally, this refers to SQL and PL/SQL code written in the implementation process and documented in Appendix A.

3.2.1 Users

Simply, a user is someone who takes part in the use of the archive system. Every single person who deals with the system is a user. A user is identified conceptually by a name, email address, password, and other voluntarily-given information such as Web page URL.

User status, hence login, is required if someone wants to make any kind of request or approve any kind of action. User status is not currently required for someone to view an document in the archive, so login is not required either. As discussed earlier, the option is left open to later require certain kinds of user status for the viewing of certain types of articles. There's more on types of articles below.

Instead of creating a new data model for users, this system relies on the ArsDigita Community System users table. There are two reason for this choice. First, the user tables are a fairly complete data model for storing demographic and personal information about people who log into the system. Second, the ArsDigita Community System, on which the archive relies, is completely intertwined with the users tables.

3.2.2 Realm types, Realms, and roles

A realm is a group of people who share particular levels of *administrative privilege*. Examples of realms might include MIT, the Laboratory for Computer Science, Dr. King's Lab, or Joe Bloggs' Individual Realm. These realms are flexible; a user can request that a realm be added to the system or that he or she be added to the realm. While users don't have to request to be part of the system, they do have to request to be part of a realm. Using an example to clarify, a student who is an employee of a particular department is not part of that department's realm *unless* he or she has administrative privileges at that level!

A realm type is a group of realms which are analogous administrative privileges for different groups of people. Realm types are, for the time being, a predetermined part of the system; there is, however, no reason to believe that this could not be made a more flexible part of a later version of the system as well. Realm types are currently really user group types in the ArsDigita Community System. Because of the recently-implemented permissions structure of the ACS, this user group functionality was extremely useful in determining permissions throughout the system. The following are the built-in realm types, basically designed at present to optimally suit the MIT administrative functions:

Service administration This level of administration is reserved for technical and librarian administrators of the archive service. It is considered to be the most privileged group of users. There are two unique characteristics conferred upon service administrators. First, they can approve other administrators, while in every other case, it is possible to approve only users who request to be part of levels *below* yours. Second, it is possible for service administrators to approve format changes (see below).

Institution administration This level of administration is reserved for users who are considered to be representatives of an institution, university, corporation, or other self-contained entity. Its parent realm is the service administration realm; this means that in order to be part of the institutional administration group, a user must have been approved by at least one service administrator.

Department administration Department administration membership is reserved for people with the "real world" authority to administer an academic or corporate department, such as a department head or course secretary. While this may indeed include several levels of bureaucracy, that is not a system-wide concern. A decision or decisions must be made by the institute-level administrators to decide on the mapping which a real organization will make to this virtual one. In order to be part of the department-level administration or to create a department, a user must have been approved by an institute-level administrator.

Group administration Group-level administration is conferred on those people who are not quite at department level, but do have the ability to administer a research group or other sub-department-level group. By definition, a group resides within a department. Therefore, one department administrator must approve a group administrator. In addition, groups can only be created or approved by department- or higher-level administrators.

Partnership administration In order to allow work to be completed by users outside of the academic hierarchy, such as two professors working intra-university, the partnership realm type is created. Because of its informality, membership must only be approved by its creators; however, creation of a partnership must be reviewed by a service-level administrator to prevent fraud. Example of fraud might include creating a partnership called "MIT (Maui Institute of Technology)" and hoping to pass for the actual institution.

Individual administration Finally, for later use, each individual is included in his or her own individual administration automatically (See Section 3.3.4).

A role is a privilege conferred on a user based on his or her membership within a realm. Roles might include administration of articles within that realm, the ability to confer membership to people within a particular realm, etc. A combination of roles is used to abstract how particular requests will be approved. This is described in greater detail below, in Section 3.3.3.

3.2.3 Articles and article types

An article is a document with its attached metadata. This means that the article itself is considered only part of the storage issue. Currently, article information is stored in a table called `archive_articles`, which houses only the metadata for the article and pointers to the file system where the differently-formatted copies of the articles are stored.

In conjunction with an article, there are article *types*. An article type is a set of articles that are approved by users of a particular realm or realms. As such, each article type has its own related approval mechanism, or set of people who need to approve the article in order for it to be considered of a particular article type.

Any article can be a member of any article type. This means that a document can be a Technical Report within a department and also a Generic Report of an entire institution. In addition, any

number of articles may belong to an article type. Thus, there is a SQL map which joins the two. When an article is added to the map, approval information is added which explains who approved the article and for what reasons. More about the approval process follows.

It is hoped, as indicated above, that membership in a particular article type will confer a sort of prestige on the article, thus increasing the editorial role of this service on the Web.

3.2.4 Formats and format conversions

A document format is just that: the format in which the document is displayed by the Web browser. This term is not specific to this system. The goal of enabling the use of many different formats, again, is to enable more people to view documents, because of personal preference and system capability.

Currently, the system can handle a few document formats, the bootstrapping formats. However, it is fully intended that users of the system will request that new formats be handled. This can be done through a standard *request*. In order to request that a new format be added to the system, the user must present more than just the identifying information about that format; he or she must also tell the system which commands to execute in order to convert to and from that format from and to the existing formats in the system.

As discussed in the next section, format requests are special in that they can be approved only be users with service-level administrative rights. At the same time, the format-conversion tools must be approved by the administrator, in order to guarantee that no malicious code will be automatically executed by the system. The overall additional burden on the system-wide administrators should not be that great, given that there are limited document formats that can be requested.

3.2.5 Approval mechanisms and decisions

An approval mechanism, as discussed briefly above, is a rule which describes a group of users, all or some of whom must approve an article in order for it to be considered part of an article type. That is, there is exactly one approval mechanism per article type, in the current implementation of the system.

An approval mechanism is currently a list of clauses. Each clause is a number and a realm; this specifies the number of people from the realm who must all approve the article in order for it to be approved. An example of a clause might be "Two people from the Carnegie Mellon Computer Science Department realm" or "One person from the Ronald Rivest realm." It is for programmatic simplicity that each user was so placed in his or her own realm, in order that each clause contains only a number and a realm, not a number and some other kind of pointer.

Currently, each of the clauses is ANDed together as follows. Taking the two clauses from the previous example, the end product would indicate that Ronald Rivest must approve the article in order for it to be considered part of the given article type. In addition, two people from within the Computer Science Department realm at Carnegie Mellon must also approve the article, although it is not important which two people they are.

For further programmatic simplicity, the "partnership" realms were created. What if a user wanted to create an article type in which either Shafi Goldwasser *or* Silvio Micali had to approve

an article? There is currently no realm to which only the two of them belong; then again, putting in each of their individual realms doesn't work either, because it creates a situation in which both Goldwasser and Micali must approve. Thus, the user requesting the above article type may create a partnership of Silvio Micali and Shafi Goldwasser. Thus the approval mechanism would include only one clause, which would read "One of the members of the partnership including Goldwasser and Micali." This would perform as previously indicated.

In order to avoid a problem of infinite recurrence, the approval mechanisms must be approved only by one person within the article type's relevant realm. Otherwise, what would the approval mechanism for the approval mechanism for the approval mechanism ... be? Thus, the system relies only on the most recently approved approval mechanism, but this can be changed and edited.

3.2.6 Decisions

What happens once an article type is defined, its approval mechanism approved, and an article uploaded for approval within that article type? That is, what if someone now comes to the site to upload a document as an "AI Lab Technical Document," which requires the approval of three members of the AI Lab faculty?

First, the site notifies, through email, all of the people in the AI Lab realm that there is a document awaiting their approval. Imagine that the email was sent to 100 people within a departmental realm, but only two people's approval was actually required. Imagine, then, that five or ten of these 100 people visit the site to read the article. Each person must make a *decision* as to whether the document meets the standards of the article type. As soon as the number of approval decisions equals the number of approvals required from each realm (clause) defined for the article type, the document is considered to be approved.

Currently, there is no way for a single person to blackball the article. There could also be the opposite, denial mechanisms, as well. There could also be definitions of interfaces between approval and denial mechanisms such that their relationship could be completely flexible; however, that is not implemented in this version of the system.

3.3 State flow

There are several paths of information flow through the archive which define the system itself. These design and implementation of these paths truly sets the groundwork for any duplicate effort. Generally, this part refers to the Tcl code written for the implementation, and is documented in Appendix B.

3.3.1 How permanent URLs are assigned

Currently, the system relies on a terribly simple algorithm for handing out permanent URLs. This basically operates as follows:

`http://archive.lcs.mit.edu/articles/id.extension`

An example of this might include several of the following PURLs:

`http://archive.lcs.mit.edu/articles/25.html` The HTML version of the article. This was the one originally input into the system.

`http://archive.lcs.mit.edu/articles/25.txt` The ASCII version of the article. Since the system knows only how to convert HTML to ASCII, this is the only other format in which this article may be available.

`http://archive.lcs.mit.edu/articles/summary/index.tcl?article_id=25` This is the status page for each article. This would include some of the pre-entered metadata as well as the status of its article type approvals. For example, "Pending approval as a Carnegie Mellon University Report." This listing will not include denials, but will include acceptances. Although details about the status will be emailed to the submittors, it will not appear here for privacy reasons. The status page will also link to the article in all of its formats as defined above.

Thus, there are two types of PURLs. Both are currently attributed to `archive.lcs.mit.edu`; this can be changed before the system goes live and has many users, so that it is more universal. The first type of PURL is the article itself, presented in different formats. The second is the summary document which presents metadata and status reports.

This PURL assignment mechanism is reasonable because a unique identifier is attached to any article uploaded into the system, and because the base URL is a permanent location for the service itself. The system hands out exactly as many pointers as necessary. When the URL is assigned to the article, the user is informed and the location stored permanently in the database.

There are certainly more interesting and readable ways for this assignment to be done. However, the most important characteristic of the system was that it needed to be a unique pointer and relatively simple to include; for this reasons, this mechanism is sufficient. Had any mechanism been chosen without a unique identifier inserted by the system, there would have been a risk of collisions.

Consequently, the state flow for parsing the PURLs is extremely simple. Basically, the decision of which URL to display depends only on two factors: its location and its extension. There are two relevant directories: the `summary` directory and the `articles` directory. Then, there is the part of the file name (before the dot) that indicates the article's unique identifier and the part of the file name that indicates the article's format.

3.3.2 How requests are made

On a more generic level, how are requests to the system made? There are two parts to this problem. The first is the collection of the appropriate data; the second is the selection of the appropriate approval mechanism, as requests and approvals are tightly linked.

All requests are made initially from the user's workspace. There are currently the following kinds of requests available, regardless of the user's administrative rights:

- Request a new article type Request that the system tolerate a new type of article, specific to a certain realm, with a certain new or existing approval mechanism.
- Request article type membership Upload an article and have it included in 0 or more article types.
- Request a new realm Request that a new realm be created.

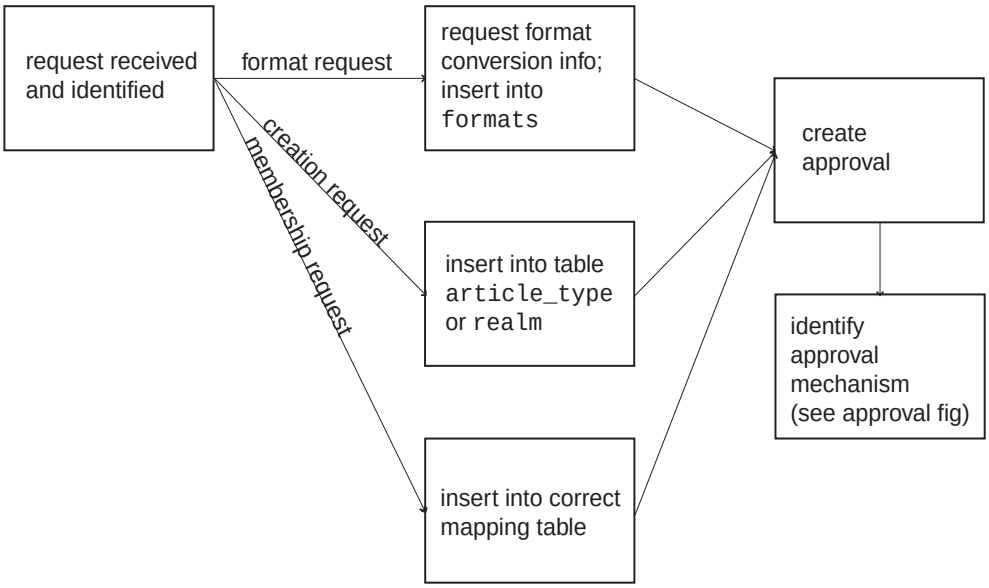

Figure 3-1: State flow for how requests are made

- Request realm membership Request that the system accept you into the membership of a particular realm.
- Request a new format Request that the system tolerate a new format, converting document formats accordingly.

Depending on the option chosen, a different input page is displayed. The data relevant to each of these requests is then ascertained. The specifics of this data entry are described in Figure 3-1. Then the data is stored as "pending" in the database and the appropriate users, as defined below, are notified of the change in the database. More information on this can be found in Appendix A.

3.3.3 How approval decisions are made

When the system receives a request, what happens next? Who does it decide to notify? It notifies everyone in the appropriate realm(s). Exactly how the appropriate realms are defined is a function of the request made, and specified as follows:

- Approve a new article type Since an article type is particular to a realm (like "Stanford University Department of Biology White paper"), the article type is approved by members of the realm to which the article applies. In this case, any member of the "Stanford University Department of Biology realm" would be allowed to make this approval. Again, this is done on a single-person basis; any other way would be too complex for the current implementation of the system.
- Approve article type membership Since an article type is mapped to an approval mechanism, the article is stored in the database and the notification is sent to the users specified by the article type's approval mechanism.

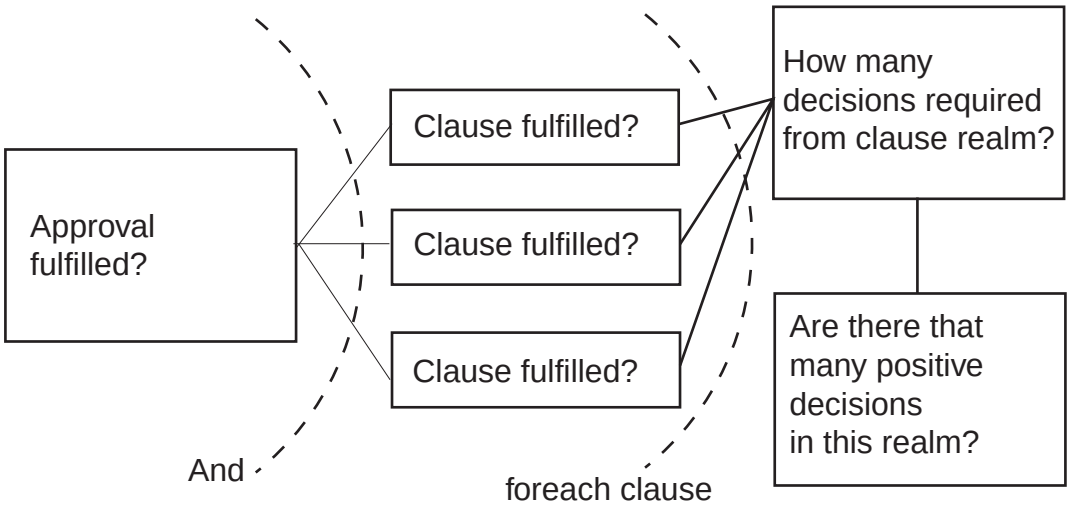

Figure 3-2: State flow for approvals

- Approve a new realm If a new realm is requested, it must be approved by one or more members of the suggested *parent* realm. For example, the General Electric "Light-bulb Division realm" must be approved by a member of the "General Electric" realm. Again, this is done on a one-or-more-user approval basis for simplicity and can later be attached to an approval mechanism without much of a problem.
- Approve realm membership Similarly, in order to be accepted into a realm, at least one user at the parent realm must accept the request.
- Approve a new format Because this is a system-wide and technical decision, to approve or deny a format and its conversions relies on the system-wide administrators' approval. This is clearly a special case because of the nature of the request.

Once the appropriate users are notified, there is nothing left to do but wait for them to come to the site, as described in figure 3-2. When they come to the site, users with administrator privileges are presented those administrative options currently awaiting their attention. They can then view the request in great detail and approve or ignore these requests with commentary.

As previously indicated, when approval is reached or the system times out (with variable time based on the approval mechanism), the user is notified.

If a user makes a request and, as defined by the above characteristics, he or she is authorized to approve his or her own request alone, then no one else is notified. The approval information is stored in the database as being his or her own, or automatic.

3.3.4 Approval mechanisms

Finally, how specifically are the approval mechanisms chosen and identified? They are input as part of the new article type request and approved and/or edited upon approval of the article type,

as indicated above.

Specifically, approval mechanisms are checked with a trigger upon each entry into the decisions table to see whether the decision has caused the approval to be complete.

As previously indicated, each of the clauses in the approval mechanism is joined by an AND clause. Thus, at an insert into the decisions table, it is first checked whether the decision has caused a clause to be fulfilled. If the clause is not fulfilled, then clearly the approval cannot be fulfilled. If the clause is fulfilled, then the other clauses are checked for similar fulfillment, and the system no longer solicits decisions from constituents within the relevant decision realm. If all of the clauses are fulfilled, then the approval is said to be made, decisions are no longer solicited, the request is marked approved, and the requesting user is notified.

Chapter 4

Potential effects of the Archive

There is a delicate balance of power between commercial journals and libraries of the academic world, a balance of power that the libraries want to change. How did this situation develop? How can the situation be changed using the archive, if at all, and why should it be? What other positive effects can be the archive add? This chapter will attempt to answer those questions.

4.1 Current situation

Commercial academic journals are currently the primary repository for ongoing academic research. And a large amount of scholarly or academic research done in the United States is completed with universities and other institutions of higher education. Journals are so important and widely-regarded, in fact, that most universities and institutions of higher education consider faculty publication within their pages to be the primary evidence of scholarly work. Indeed, it is frequently, though controversially, said that writing published in journals should "be considered whenever judgments of scholarly achievement are being made." [19]

If and when most young faculty members at research-oriented universities in the United States face all-important tenure decisions, they will have to have published work in respected and referenced journals. Given that tenure is the primary indicator of indefinite job security for those in higher education, it is literally impossible to overestimated its importance. Therefore, the importance of publication, both within academic journals and without, is of paramount importance. While textbooks and other forms of publication are of course considered viable forms of output for scholarly work, journals are truly the only place to publish relatively short, frequent updates for peer review and recognition. This is especially true during those first few, pivotal years of academic employment.

On the other end, what happens when researchers at any level or in any walk of life who wish or need to use the work published in these commercial academic journals? What happens when the sixteen-year-old in Alabama or any other rural locale is learning about biology and wants to learn past the ordinary high school level? What happens when a college student is learning about medieval French literature and to learn more about modern study on the subject, but that sort of work isn't readily available at her university? Usually, this is the time when local or university libraries become involved in the research process.

Libraries currently provide archives not only of books but also of reference works, including

commercial journals. Using these archives, researchers at all levels of general or particular interest are able to find research relevant to their interests. Books on the topic may be published months or years after monthly research articles, because the publishing of books has more costly overhead than the publishing of academic journals, and takes longer in addition. Thus, libraries have taken the role of facilitators, connecting people with the appropriate academic journals.

> It was once a buzz in the hallways of academic conferences, library association meetings, and scientific forums. But during the past decade, the journals pricing crisis has become a roar echoing around the world. Although there has been considerable debate about how best to transform scholarly publishing, no solutions have yet been so widely embraced that the underlying structures have shifted appreciably. There's a good reason for this: the issues involves unprecedented technology shifts, economic forces that have been a generation in the making, and the heterogeneous cultures of scientific communities and academic institutions. [9]

Unfortunately, this is a very expensive position they have taken, as Johnson indicates above. Library subscriptions to these journals are very expensive. In the early 1990s, "America's 3,500 academic libraries [spent] more than $1.25 billion a year on acquisitions" [15]. In the late 1990s, the Massachusetts Institute of Technology Libraries alone spend about $2.4 Million per year on subscriptions to nearly 6,000 academic journals; some are free, but most are not [3]. "Like research libraries worldwide, [Denmark's national Technical Knowledge Center and Library] has been badly hit by the spiraling inflation of journal prices that has resulted in libraries paying more to provide users with less"[2]. In addition, journal publication costs have risen dramatically faster over the last three decades than other "other measures of national growth," according to Okerson [15].

However, while this may make sense from the financial standpoint of academic journals, it is prohibitively expensive from the standpoint of the libraries. Because, fundamentally, libraries are non-profit or poorly-funded organizations. Many local public libraries, for example, are particularly poorly funded. Thus, they may lack in the important research available to better funded libraries but still important to the education of their constituencies.

In addition, the libraries of the world do take issue with the current control situation. They see university libraries as somewhat more altruistic than their commercial counterparts. Any effort in the form of an online archive which would "move scholarly publishing in from the busy commercial street and closer ... to home, ... will have accomplished something worth our while," according to Ann Okerson of the College and Research Libraries [16]. Commercial journals really decide who publishes what, and most academic journals are for-profit institutions. They have sponsors and advertisers. The *Journal of the American Medical Association*, for example, accepts advertisers from fields related to the research it presents. Although it lays out an elaborate policy document called the "Principles Governing Advertising in AMA [American Medical Association] Publications" [1], it is impossible to truly assess the effect of advertising on the content of academic journals. And the objectivity of choice of that content has indeed been called into question on occasion.

As well, paper copies of these journals are growing less appealing to researchers, who find digital copies of these journals easier to peruse [2]. While the most efficient way of finding articles relevant to a search was once to look at related titles in journal archives, that is no longer the case. An online service could provide access to even more information, searching through abstracts and even

providing useful summarization tools for each article. Thus, what used to be the fastest and most useful format, paper journals in related archives, may no longer be so useful.

Recent changes to copyright law only exacerbates the tension of this already-strained relationship. This aggravation comes in the form of the Digital Millennium Copyright Act of 1996. This act gives increased additional control over intellectual property to its owners. But who are the owners of the majority of published research? Arguably, this control may fall to the commercial academic journals. Since journals have the virtual monopoly over scholarly publishing described earlier, there is no reason to believe that they couldn't demand retention over most of the intellectual property rights, even if those rights are greatly expanded. However, libraries have been granted "modest" exemption from these laws but important, because they will "make [the libraries'] task of maintaining collections easier in a technologically advanced environment" [14].

The exemption may allow libraries to use different techniques to either confer intellectual property onto the actual authors of the research. First, the libraries can simply use their exemption to relay online access to articles. Second, the libraries, in acting like an independent electronic journal, could actually allow their user-customers to take back control over intellectual property that the users created. This exemption, used with a technological system, may provide the libraries with some much-needed strength.

4.2 Code as factor for change

> More and more, those with interests to protect (copyright holders, for example) or interests to exploit (feeders on personal data) will use code writers of Silicon Valley rather than Congress to secure their interests. [11]

Why use a technological system for change? This is best described by Harvard Law School Professor Lawrence Lessig's *Law of the Horse* [11], as quoted above. He describes that there are four ways to regulate behavior, as we wish to excise commercial journals from the life of researchers. The first is through *law*; libraries could lobby government to pass regulation which favors non-profit organizations over the use of commercial journals. The second is through *social norms*; libraries could create public service campaigns begging researchers and academics not to judge tenure cases or academic achievement based on publication in journals which are for-profit. Third, libraries could use *markets*, accepting sponsorship from professional and related organizations so that that would no longer be at a monetary disadvantage to commercial journals. Fourth and finally, libraries could rely on *code*, creating archives of articles meant to disintermediate commercial journals from the scholarly publication process and hope that, through its availability, it will be used. This paper presents one option in the move towards greater library involvement in the academic research and publication process: a technological system for commercial journal disintermediation in academic publishing.

Put simply, the system is an archive which stores research articles and some cataloging information about them. When an article is uploaded by a user, and approved by the appropriate faculty members if necessary, the system provides the author(s) with a permanent pointer to the article so that he or she can use it confidently as a reference point for other researchers to find. It may provide other functionality, such as converting the articles into other formats for use by people with different functional capabilities. It may also allow people with different roles, such as faculty

members or librarian-administrators, to play those roles entirely through the system.

That is why this system was designed: to provide a code-based solution to a socio-economic problem (that is, "political" with a lower-cased p). As corroborated by the previous examples, technological solutions are frequently more feasible, less expensive, and more effective than their legal, social, and market-based counterparts. And so we begin on this voyage.

What is the goal here? Ideally, every researcher in the world would choose to use this system or similar online journal systems, rather than publishing their articles in for-profit, advertiser-supported, subscription-required journals. Every researcher would submit his or her articles, upon completion or during development, to the archive. He or she would then put the approved, permanent URL on his or her *curriculum vitae* online, sending email or other notification to peers to begin a peer review of the work. Notoriety would spread through the use of the system, and tenure committees would consider relevant approval of one's article to be as prestigious an honor as publication within a commercial journal is today. Libraries would be brought into the loop, commercial journals more or less kept out; the researcher would retain greater control over his or her work, and intellectual property would belong to the intellectual.

In the worst case scenario, an archive of this type does no harm. Even used in only minimally, it provides *curriculum vitae* to researchers who are interested; since the marginal cost of administration is near zero, then it doesn't truly matter how many people use the system. At worst, it may provide enough attention from the commercial journal world to increase competition and possibly lower some outrageous subscription costs.

4.2.1 Other technological systems for political change

As indicated by Professor Lessig's *Industry Standard* article as quoted above, the idea to employ code for change in a political situation is not an entirely new one. Rather, it is proving more and more useful, as corroborated by these other systems built at least in part to outsmart a bad non-technical situation.

As early as 1990, the idea of "electronic journals" was being discussed as an alternative to expensive journals, and to provide value to libraries worldwide. Okerson's "With Features" article makes direct mention of these "journals," but she refers to the technical "lag" because of the need to transmit "non-text data" [16]. In some sense, this situation has changed. With the widespread use of the Web, this lag no longer really exists!

Thus, some electronic journals have indeed been built. Within the scholarly publishing world, there are two eminent examples of similar archiving systems. The e-Print Archive at the Los Alamos National Laboratories houses thousands of articles related to all areas of physics research [10]. The High Wire Press, founded at Stanford University, also provides helps academic groups publish at lower than market costs [12]. These two systems were designed and built for primarily the same reasons that this one is.

4.2.2 Why this idea is different

Research article archives have existed in many formats, and exist on the Internet, as described above. This one, however, brings together features that haven't been used before. It is intended

that these features will provide additional usability to the system, reduce the overhead that using it will cause, and increase individual, rather than organizational, power over the intellectual property.

4.2.3 Realms provide automated administration

The realms provided by the system mean that, outside of bootstrapping and format approvals, the administrators of the system can allow the individual article types to be controlled by their related realms. According to Eric Celeste of the MIT Libraries, this is not something done in the current, electronic and non-electronic systems within the MIT Libraries [3]. Certainly, use of this feature would greatly reduce administrative overhead for the librarians of the system, because far fewer approvals would go through them than ordinarily do.

Permanent pointers increase stability of the Web

When a researcher publishes in a commercial academic journal, he or she is completely aware that the work will be archived and available in its final form practically indefinitely. This will be done through both paper archiving and standard cataloging techniques, but also through microfilm and/or microfiche, and on digital archive in certain locations. But that sort of stability doesn't necessarily exist in any current archiving system.

Two of the archive's main features are that it increases the *perceived* stability of the Web as a publishing mechanism, and reduces storage demands on the part of the author. Stability is a reasonable concern, since an upsetting percentage of the links generated by a search engine or other online service are broken ones that point to nothing, indicative of a very fast-moving and immature medium.

The service will house the article's data and metadata (data about the article, such as title) in storage on the server(s) it uses. There are other options. One appealing option is to provide only a service which allows a researcher to get metadata about his or her article, load a pointer to the researcher's local copy online, and have people point it there. This would still provide some of the desired stability by providing a permanent pointer to the work. It will reduce the load on the server storage and access requirements, because it doesn't require that storage be done centrally.

However, by requiring the user to retain control over the storage of the work, it increases their responsibility over archiving their publications. As indicated earlier, the archive must reduce (or, at worse, maintain the *status quo* of) researcher overhead in order to increase archiving outside of the realm of for-profit publications. Thus, the solution chosen in use with by system will be a combination of the permanent pointer as described above and the central storage of the archive data and metadata. This will provide not only the desired stability discussed earlier, but will dramatically reduce overhead required of authors; they will simply upload their articles and forever know that their article is safe and available indefinitely.

Approval mechanisms increase quality of content

It seems to be accepted dogma that print journals will always exist, because they provide some "guarantee of quality and added editorial value" [2]. An additional, complimentary argument is provided by newspaper editors about the appealing feel of the news print, by book authors about the mass of the book and the beauty of the printed photographs relative to their online counterparts. While those arguments may be true, it is not necessarily the case that a printed

document is necessarily the product of more stringent editing. Although the time invested may be greater for a printed journal, newspaper, or book, it is possible to apply editorial functionality to online work. Indeed, most editors of real-world reading material currently use technological tools to their advantage [7]. Technological tools can certainly be used by editorial people to provide a similar level of quality with even less overhead and lower time to production.

The archive system will provide the ability for appropriate administrators, such as tenured faculty members, to approve articles for use in the system, based on the level of approval which the author seeks. This will, it is hoped, counteract the notion that all Web content is completely devoid of editorial value and guarantee. It is simply held that if the reputation of the archive, in representing a university, can reach the reputation of a commercial academic journal, possibly even surpassing it. Additionally, since a certain piece of research is approved only to a particular level ("This article has been approved by three tenured faculty members as a technical report of this laboratory" or "This article has not yet been approved for use in this archive"), that provides yet another level of granularity for certifying the quality of that article.

Furthermore, the system has flexible approval facilities, which will provide possibly *greater* editorial control. It is possible for an academic or librarian-administrator to create a new type of article type, such as "Whitehead Institute Technical Report" and create an approval process used only for this type. An example in this case could be: "Any three tenured Whitehead faculty and one Whitehead administrator must approve this article before it is considered a Whitehead Institute Technical Report." This sort of grouping can be done at the Institute, school, department, laboratory, group, or even individual level; it is intended to be flexible!

Flexibility allows for wider, more long term use

The article-type flexibility just described is only one of the ways in which the service will be designed for evolution within the library and academic communities. This is an important goal because obsolescence of stored materials is a real problem for libraries [21], and it is to be avoided for as long as possible.

There are many examples of this desired flexibility. Besides the different approval mechanisms, the article types provide a way in which many distinct, currently-existing archives could be provided through the use of this one. That is, by selecting "Physics research document" when uploading an article, and allowing specific physics department approval of those documents, the system has recreated the Physics archive provided at the Los Alamos National Labs.

In addition, users and administrators could request that additional document formats be added to the system to provide more utility to people within their field. For example, many researchers at the MIT Media Laboratory rely on Microsoft Word as a text editor and exchangeable document format. But researchers at the Laboratory for Computer Science frequently prefer LaTeXas a publication tool and document format, so having a document in Word format would be practically useless, especially if they are running an operating system which truly doesn't allow them to read those documents. Thus, a researcher at LCS could introduce new document formats and explain to the system to do conversions from the existing formats, thus opening up the service's utility to the other researchers within his or her lab. And these are just some of the flexibility goals that the system has.

Common metadata formats allows integration with other systems

There is one point of flexibility so important as to deserve its own heading: interoperability with other systems. By trying to integrate standard import and export metadata into the archive, the system's design may allow it later to conveniently deal with other archives or searchable indices. If there is a useful physics database, for example, the searching facility within this general archive can query the physics archive for information through the shared use of a common metadata format. Also, if there is another general library archive at another institute of higher learning or public library, these archives can be made interoperable using this metadata exchange. And through interoperability, once again, obsolescence is avoided and utility is maximized!

These were the hopes of the system: that their features would encourage a change in the *status quo*, allow libraries some useful features, and encourage the use of the archive and systems like it by researchers worldwide. There are signs that this may happen: according to Okerson, by 1990 there were "a handful of innovative risk-takers were ready to commit to a new type of [electronic] journal" [15]. Nearly ten years later and with the increased commercial availability of the Internet, this has, it is hoped, moved from a handful of innovators to a majority of university researchers.

Chapter 5

Conclusions

Having considered, designed, and built an archive for scholarly publishing, two questions remain. The first: "What remains undone?" The second: "What has been learned?"

5.1 Future work

The future work described here is focused on large-scale changes within a small-scale realm: feature additions or modifications to the archive which drastically alter its purpose or use.

5.1.1 Enhanced system security

The system, currently at the prototype stage, relies on user identification through login by email address, and does very little checking for truth in advertising. However, there are two steps which could be taken – and should – in order to increase the security of the system.

The first option is one implemented by the Virtual Compassion Corps [6], another identity-reliant Web service: ask the user to log in using email address. Rather than asking the user for a password and trusting that he or she is telling the truth about his or her email address, the user is e-mailed a randomly selected password. In order to log in, the user must provide the e-mailed password, thereby verifying that the email address is really his or hers.

The second option, which is more secure but less universally scalable, is the use of digital certificates for individual identification. Digital certificates are desirable for two reasons, aside from their inherent security. First, they are already ingrained into the Institute's daily life, required for frequently-used services such as WebSIS [18]. When dealing with security, the more ingrained a system is, the better; otherwise, it is a hassle and people tend not to use it. Second, AOLserver will soon support digital certificates, allowing the present architecture to tolerate a great security option.

As indicated previously, neither of these options is currently implemented because security was not considered crucial to the point of the prototype. However, these two solutions are the recommended security options once the system goes into production.

5.1.2 More personalization

Currently, the personalization on the site is basically limited to the extent that each use is presented only with options that he or she is allowed to perform. So if a user is a generic writer, but has no administrator privileges of any kind, he or she can view articles, request an upload of an article, or request any number of permissions.

However, there is currently very little in the way of personalized information; content, not choice, which is different for each individual using the system. One example of this might include bibliographic bookmarks, as outlined in Chapter 2.2 but not implemented. Another example might include relevant news information posted to the user's workspace within the site. Yet another example might include providing the user with useful defaults; for example, providing default keyword options or collaborator names based on prior works stored in the system. Put together, this would add up to increasingly personalized service, allowing increased return on decreased researcher investment of time and work.

5.1.3 Better navigation

Currently, the navigation of the site is limited to requests made on the part of users and approvals granted or denied by the relevant administrators and the searching functions implemented. Articles are linked into the user interface of the archive only through the pURLs passed out by the system.

A certain amount of hierarchical data organization is a welcome introduction into a system that contains a large amount of content. This theory is corroborated by popular yet content-intensive sites like Yahoo! [23] or the Los Alamos e-Print Archive [10]. Ideally, the keywords and group-specific information provided by authors, and approved accordingly by administrators and faculty members, can be used in later versions of the system to populate an arbitrarily created ontology for navigation of data. Once again, this would provide another useful mechanism for reducing researcher overhead.

5.1.4 Extended collaboration among authors

The site is, primarily, intended to help researchers and research institutions. This is done through two mechanisms: first, it allows authors to publish in a recognized forum, outside of the commercial journals currently available; second, it allows users to collaborate on work by helping them read published research without paying for subscriptions.

However, the site could promote collaboration among researchers and other authors by allowing editing to be done in its midst. If the site could be used as a version control system, researchers disparate in time and space could work together on ongoing work, even allowing for peer review – if desired – to happen at different stages in the writing process. This would certainly be a useful tool for researchers, as well.

5.1.5 Extended collaboration among archive services

As indicated in several sections, collaboration among users is not the only desirable form of collaboration. Allowing services on the Internet to share their wares would certainly increase the scope of each researcher's work, whether the work was the posting of information or the search for more information.

As indicated in the previous chapter, some effort is made to incorporate standard metadata tag sets into the system for inter-site collaborative purposes. However, any way that this could be increased or expanded would provide additionally powerful collaboration tools. As XML [22] and RDF [5] become more and more widespread tools for metadata definition, they will be more reliable as tools for interoperability among sites and should be even further integrated into its use and development.

5.1.6 More active community

The site should move towards an "online community" focus in order to provide increased collaboration in other aspects of the system's use, in addition to the collaboration between sites and among users for creation and manipulation of documents and their metadata. In this vein, users in a bulletin-board or commentary forum could provide feedback on articles. In this way, the archive would provide realtime and public peer review of articles. In addition, it would open up a real dialog between authors and more established experts in the field. Finally, it might allow less experienced researchers, such as college or high school students, to ask questions at different levels of expertise. All of these options would add significantly to the value of the archive, and also increase tremendously the competitive advantage of this service over the existing commercial journal infrastructure, and should be incorporated into production of this archive.

5.1.7 Integration with other systems

Finally, the MIT libraries do not currently use our prototyping tools, such as Oracle and AOLserver. Although, as described in Chapter 3.1, they are good tools for Web development and especially for prototyping, the service will not be useful if it cannot be integrated at all into the existing libraries technology interface. Thus, when the system is to move from proof-of-concept and initial design to fully used system, either the libraries must adopt these technologies, the system must be altered to suit the technology preferences of the libraries, or they must meet in the middle.

5.2 Proof of concept and work completed

Archives such as this one show that it is possible to create technological systems which can, through widespread use, work around established, negative situations. And, if a system is to be designed and built whose sole purpose is to change an existing balance of power relationship, it must be done with an extremely strenuous set of criteria. In short, the technological system must provide service at a level far above the *status quo* in order to achieve that widespread use, and must do so for all of the interested parties.

In addition, as a service, the archive prototype has gone beyond other library services in terms of usability and system flexibility. It provides end-user and administrative features which make the service more appealing to researchers such that they have a viable alternative to expensive, monopolistic academic journals. In addition, the system's extensibility allows it to evolve with time to changing needs of the library world and of researchers involved in publishing communities of all kinds.

Appendix A

Data model

```
--
-- Articles
--

create sequence asp_article_sequence;

create table asp_articles (
        article_id              integer not null primary key,
        other_identifier        varchar(200),
        source                  varchar(200),
        title                   varchar(2000) not null,
        authors                 varchar(2000),
        keywords                varchar(2000),
        version                 varchar(200),
        submit_date             date,
        create_date             date,
        description             varchar(4000),
        base_location           varchar(1000),
        permanent_url           varchar(1000),
        creator_id              integer not null references users
);

--
-- Approval mechanisms
--

create sequence asp_mechanism_sequence start with 5;

create table asp_mechanisms (
        mechanism_id                integer not null primary key,
        mechanism_name              varchar(200)
);

insert into asp_mechanisms
```

```
(mechanism_id, mechanism_name)
values
(1, '');

--
-- Clauses
--

create sequence asp_mechanism_clause_sequence;

create table asp_mechanism_clauses (
        mechanism_clause_id             integer not null primary key,
        mechanism_id                    integer references asp_mechanisms,
        mechanism_clause_n_people       integer,
        mechanism_clause_group          integer references asp_realms
);

--
-- Approvals
--

create sequence asp_approval_sequence;

create table asp_approvals (
        approval_id                      integer not null primary key,
        approval_date                    date,
        approval_name                    varchar(400),
        approval_mechanism_id            integer references asp_mechanisms,
        approved_p                       char(1) check (approved_p in ('t','f'))
);

insert into asp_approvals
  (approval_id, approval_date, approval_name, approval_mechanism_id, approved_p)
  values
  (asp_approval_sequence.nextval, sysdate, '', 1, 't');

--
-- Article types
--

create sequence asp_article_type_sequence;

create table asp_article_types (
        article_type_id                 integer not null primary key,
        article_type_name               varchar(200),
        article_type_realm_id           integer references asp_realms,
        article_type_suggestor_id       integer references users,
```

```
        article_type_approval            integer references asp_approvals
);

create or replace view asp_approved_article_types as
 select *
 from asp_article_types
 where 't' = (select approved_p
              from asp_approvals
              where approval_id = asp_article_types.article_type_approval);

create or replace view asp_unapproved_article_types as
 select *
 from asp_article_types
 where '' = (select approved_p
            from asp_approvals
            where approval_id = asp_article_types.article_type_approval);

--
-- Relate articles to article types
--

create table asp_article_type_map (
        article_id                integer references asp_articles,
        article_type_id           integer references asp_article_types,
        approval_id               integer references asp_approvals,
        primary key (article_id, article_type_id)
);

create or replace view asp_approved_at_map as
 select *
 from asp_article_type_map
 where 't' = (select approved_p
              from asp_approvals
              where approval_id = asp_article_type_map.approval_id);

create or replace view asp_unapproved_at_map as
 select *
 from asp_article_type_map
 where '' = (select approved_p
             from asp_approvals
             where approval_id = asp_article_type_map.approval_id);

--
-- Decisions
--

create sequence asp_decision_sequence;
```

```
create table asp_decisions (
        decision_id             integer not null primary key,
        approval_id             integer references asp_approvals,
        decision_date           date,
        decider_id              integer references users,
        decision_p              char(1) check (decision_p in ('t','f')),
        decision_description    varchar(4000)
);

create or replace function asp_clause_fulfilled_p
   (v_n_people in INTEGER, v_group in INTEGER, v_approval_id in INTEGER)
RETURN char
AS
        clause_count         INTEGER;
BEGIN
        select count(*) into clause_count
        from asp_decisions
        where approval_id = v_approval_id
        and decision_p = 't'
        and exists (select user_id
                    from asp_realm_user_map
                    where realm_id = v_group
                    and user_id = asp_decisions.decider_id);

        if clause_count >= v_n_people
        then return 't';
        else return 'f';
        end if;
END asp_clause_fulfilled_p;
/
show errors;

create or replace function asp_approval_fulfilled_p
    (v_approval_id in INTEGER, v_mechanism_id in INTEGER)
RETURN char
AS
        approval_count INTEGER;
BEGIN
        select count(*) into approval_count
        from asp_mechanism_clauses
        where 'f' = asp_clause_fulfilled_p(mechanism_clause_n_people,
                                           mechanism_clause_group,
                                           v_approval_id)
        and mechanism_id = v_mechanism_id;

        if approval_count = 0
```

```
        then return 't';
        else return '';
        end if;

END asp_approval_fulfilled_p;
/
show errors;

--
-- Formats and format conversions
--

create sequence asp_format_sequence;

create table asp_formats (
        format_id               integer not null primary key,
        format_extension        varchar(10) not null,
        format_name             varchar(200),
        format_suggestor        integer references users,
        format_approval         integer references asp_approvals
);

create or replace view asp_approved_formats as
 select *
 from asp_formats
 where format_approval in (select approval_id
                           from asp_approvals
                           where approved_p = 't');

--
-- Conversions
--

create table asp_format_conversions (
        source_id                   integer references asp_formats,
        dest_id                     integer references asp_formats,
        conversion_commands         varchar(400),
        conversion_approval         integer references asp_approvals,
        primary key (source_id, dest_id)
);

--
-- Realm types
--

create sequence asp_realm_type_sequence start with 10;
```

```
create table asp_realm_types (
        realm_type_id             integer not null primary key,
        realm_type_name           varchar(400)
);

-- System

insert into asp_realm_types
(realm_type_id, realm_type_name)
values
(1, '');

insert into asp_realm_types
(realm_type_id, realm_type_name)
values
(2, 'Institution');

insert into asp_realm_types
(realm_type_id, realm_type_name)
values
(3, 'Department');

insert into asp_realm_types
(realm_type_id, realm_type_name)
values
(4, 'Group');

--
-- Realms
--

create sequence asp_realm_sequence;

create table asp_realms (
        realm_id             integer not null primary key,
        realm_parent         integer references asp_realms,
        realm_name           varchar(400),
        realm_type           integer references asp_realm_types,
        realm_requester      integer references users,
        realm_approval       integer references asp_approvals
);

create or replace view asp_approved_realms as
 select * from asp_realms where
 realm_approval in (select approval_id
                    from asp_approvals
                    where approved_p = 't');
```

```
insert into asp_realms
(realm_id, realm_parent, realm_name, realm_type, realm_approval)
values
(realm_id, '', 'System realm',
        (select realm_type_id
         from asp_realm_types
         where realm_type_name = ''),
        (select approval_id
         from asp_approvals
         where approval_name = ''));

--
-- Realm membership
--

create table asp_realm_user_map (
        user_id                 integer references users,
        realm_id                integer references asp_realms,
        approval_id             integer references asp_approvals,
        primary key (user_id, realm_id)
);

create or replace view asp_approved_realm_user_map as
 select *
 from asp_realm_user_map
 where 't' = (select approved_p
              from asp_approvals
              where approval_id = asp_realm_user_map.approval_id);

create or replace view asp_unapproved_realm_user_map as
 select *
 from asp_realm_user_map
 where approval_id in (select approval_id
                       from asp_approvals
                       where approved_p is null);


--
-- How to map approval mechanisms to particular actions:
--

create table asp_mechanism_map (
        table_name        varchar(30),
        column_name       varchar(30),
        column_value      integer,
        mechanism_id      integer references asp_mechanisms
```

```
);

insert into asp_mechanisms
(mechanism_id, mechanism_name)
values
(asp_mechanism_sequence.nextval, 'System request: requires approval of one system user');

insert into asp_mechanism_clauses
(mechanism_clause_id, mechanism_id, mechanism_n_people, mechanism_group)
values
(asp_mechanism_clause_sequence.nextval ,asp_mechanism_sequence.currval, 1, 2);

insert into asp_mechanism_map
(table_name, column_name, column_value, mechanism_id)
values
('asp_realms', 'parent_realm', '', asp_mechanism_sequence.currval);

insert into asp_mechanism_map
(table_name, column_name, column_value, mechanism_id)
values
('asp_formats', '', '', asp_mechanism_sequence.currval);

--
-- For users table, see software.arsdigita.com
--
```

Appendix B

Service-specific functional code

This section the Tcl code written to present HTML pages to the users of the system. These procedures were pasted together and form the bulk of the interesting programming for the site. The following is presented in alphabetical order by module, for reference. In addition, there are two important procedures for each module:

Print This procedure prints the HTML form for a particular module. It is used to abstract out the HTML writing portion of the system. When used for data entry, it is used alone. When used to display information from the database, it is combined with a Tcl procedure called `bt_merge_piece` which does a careful combination of the form fields with the database information.

Request Takes the information uploaded into forms and inserts the related request into the database. Always creates a new approval with `approved_p` equal to null. This allows the later procedures to check if the approval is fulfilled. If the request is the creation of something, like a realm or article type, it also creates a new row in the `asp_mechanism_map`; this row later defines how the request is approved.

These are implemented for each module when necessary; otherwise, as in `decision`, it can be implemented without regard to module.

B.1 Approvals

```
proc archive_user_approval_possible_sql {user_id} {
    # Returns all of the open approvals which the user
    # is elligible to decide about.

    return "select approval_id as this_approval_id
    from  asp_approvals
    where approved_p is NULL
    and 0 = (select count(*)
             from asp_decisions
             where asp_decisions.decider_id = $user_id
             and asp_decisions.approval_id = asp_approvals.approval_id)
    and 0 < (select count(*) from asp_approved_realm_user_map
             where user_id = $user_id
```

```
            and realm_id in (select mechanism_clause_group
                             from asp_mechanism_clauses
                             where asp_mechanism_clauses.mechanism_id =
                             asp_approvals.approval_mechanism_id))"
}
```

B.2 Articles and article types

```
proc archive_article_type_request {type_id type_name \
       type_realm_id type_suggestor_id approval_id mechanism_id} {
    # Insert the request into the database.

    set db [ns_db gethandle subquery]

    ns_db dml $db "insert into asp_mechanism_map
    (table_name, column_name, column_value, mechanism_id)
    values
    ('asp_article_type_map', 'article_type_id', '$type_id', $mechanism_id)"

    ns_db dml $db "insert into asp_approvals
    (approval_id, approval_mechanism_id)
    values
    ($approval_id, $mechanism_id)"

    ns_db dml $db "insert into asp_article_types
    (article_type_id, article_type_name, article_type_realm_id,
       article_type_suggestor_id, article_type_approval)
    values
    ($type_id, '$type_name', $type_realm_id,
       $type_suggestor_id, $approval_id)"
}

proc archive_article_type_print {article_type_id editable_p} {
    # Print article type.

    set return_string ""

    append return_string "
    <INPUT TYPE=HIDDEN NAME=article_type_id VALUE=$article_type_id>

    Article type name: <BR>
    <INPUT TYPE=TEXT SIZE=50 NAME=article_type_name>
    <P>
    Associated realm: <BR>
    <SELECT NAME=article_type_realm_id>
    <OPTION VALUE=\"\"> Select a realm
    "
```

```
    set sub_db [ns_db gethandle subquery]

    set selection [ns_db select $sub_db "select realm_id, realm_name
    from asp_realms
    order by upper(realm_name)"]

    while {[ns_db getrow $sub_db $selection]} {
        set_variables_after_query
        append return_string "<OPTION VALUE=\"$realm_id\"> $realm_name\n"
    }

    append return_string "
    </SELECT>
    <P>

    "

    if {[string compare $editable_p t] == 0} {
        # If it is being edited or initially requested, show the submit button.
        append return_string "<INPUT TYPE=SUBMIT NAME=Submit>\n"
    }

    ns_db releasehandle $sub_db
    return $return_string
}

#
# Request and approve article type membership.
#

proc archive_article_type_membership_request {article_id article_type_id} {
    # Place into the archive article_type map.

    set db [ns_db gethandle subquery]

    set approval_id [database_to_tcl_string $db "select
    asp_approval_sequence.nextval
    from dual"]

    set mechanism_id [database_to_tcl_string $db "select mechanism_id
    from asp_mechanism_map
    where table_name = 'asp_article_type_map'
    and column_name = 'article_type_id'
    and column_value = '$article_type_id'"]

    ns_db dml $db "insert into asp_approvals
```

```
    (approval_id, approval_mechanism_id)
    values
    ($approval_id, $mechanism_id)"

    ns_db dml $db "insert into asp_article_type_map
    (article_id, article_type_id, approval_id)
    values
    ($article_id, $article_type_id, $approval_id)"

    ns_db releasehandle $db
}

proc archive_article_type_membership_print {article_id editable_p} {
    # Prints the article_type_membership

    set db [ns_db gethandle subquery]

    set return_string ""

    if {[string compare $editable_p t] == 0} {
        # We're editing, so don't show the approved ones.
        set sql "from asp_approved_article_types t, asp_realms r
        where t.article_type_realm_id = r.realm_id"

    } else {
        # Not editable. We're viewing, so show the others!
        set sql "from asp_approved_article_types t,
            asp_realms r, asp_approved_at_map m
        where t.article_type_realm_id = r.realm_id
        and m.article_type_id = t.article_type_id
        and m.article_id = $article_id
        order by upper(r.realm_name), upper(t.article_type_name)"
    }

    set selection [ns_db select $db "select
    t.article_type_id, t.article_type_name, r.realm_name
    $sql"]

    set article_type_count 0

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query
        incr article_type_count

        append return_string \
        "<INPUT TYPE=CHECKBOX NAME=article_types VALUE=$article_type_id>
        [string toupper $realm_name]: $article_type_name <BR>\n"
```

```
    }

    # What if there are no article types that they can apply for?
    # If there are, hit submit. If not, tell them.

    if {$article_type_count == 0} {

        if {[string compare $editable_p t] == 0} {

            append return_string "There are currently no article types
            you can apply for.  You can request one from your <A
            HREF=home.tcl> workspace</A>.\n"

        } else {

              append return_string "This article is approved for no
              particular article types.\n"

        }

    } elseif {[string compare $editable_p t] == 0} {
        append return_string "<P> \n<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    }

    return $return_string
}
```

B.3 Decisions

```
proc archive_decide {approval_id decider_id decision_p decision_description} {
    # Inserts the decision into the database.

    set db [ns_db gethandle subquery]

    set decision_id [database_to_tcl_string $db "select
       asp_decision_sequence.nextval from dual"]

    ns_db dml $db "begin transaction"

    # Note the decision.

    ns_db dml $db "insert into asp_decisions
    (decision_id, approval_id, decision_date,
       decider_id, decision_p, decision_description)
    values
    (asp_decision_sequence.nextval, $approval_id, sysdate,
```

```
       $decider_id, '$decision_p', '$decision_description')"

    # KEY: Does this decision mean that the approval is fulfilled?

    ns_db dml $db "update asp_approvals set
    approval_date = sysdate,
    approved_p = asp_approval_fulfilled_p($approval_id, approval_mechanism_id)
    where approval_id = $approval_id
    and asp_approval_fulfilled_p($approval_id, approval_mechanism_id)
       is not NULL"

    ns_db dml $db "end transaction"
}

proc archive_decision_print {decision_id approval_id editable_p} {
    # Print a decision.

    set return_string ""

    append return_string "
    <INPUT TYPE=HIDDEN NAME=approval_id VALUE=$approval_id>
    <INPUT TYPE=HIDDEN NAME=decision_id VALUE=$decision_id>
    <P>
    Approved? <BR>
    <INPUT TYPE=RADIO NAME=decision_p VALUE=t> Yes <BR>
    <INPUT TYPE=RADIO NAME=decision_p VALUE=f> No
    <P>
    Decision comment: <BR>
    <TEXTAREA ROWS=10 COLS=80 NAME=decision_description>
    </TEXTAREA>
    <P>\n"

    if {[string compare $editable_p t] == 0} {
        append return_string "<INPUT TYPE=SUBMIT NAME=Submit>\n"
    }

    return $return_string
}
```

B.4 Formats and format conversions

```
proc archive_format_print {enabled_p} {

    append return_string ""

    set db [ns_db gethandle subquery]
```

```
    set selection [ns_db select $db "select
    format_id, format_name, format_extension
    from asp_formats
    where format_approval in (select approval_id
                              from asp_approvals
                              where approved_p = 't')"]

    set return_string "<SELECT NAME=format_id>\n"

    while {[ns_db getrow $db $selection]} {
       set_variables_after_query

       append return_string "<OPTION VALUE=$format_id>
       $format_name ($format_extension)\n"
    }

    append return_string "</SELECT>
    <P>\n"

    if {[string compare $enabled_p t] == 0} {
        append return_string "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    }

    ns_db releasehandle $db

    return $return_string

}

proc archive_format_request {format_id format_extension \
       format_name approval_id} {

    set db [ns_db gethandle subquery]

    set mechanism_id [database_to_tcl_string $db "select mechanism_id
    from asp_mechanism_map
    where table_name = 'asp_formats'"]

    ns_db dml $db "insert into asp_approvals
    (approval_id, approval_mechanism_id)
    values
    ($approval_id, $mechanism_id)"

    ns_db dml $db "insert into asp_formats
    (format_id, format_extension, format_name, format_approval)
    values
```

```
    ($format_id, '$format_extension', '$format_name', $approval_id)"

    ns_db releasehandle $db
}

proc archive_format_conversion_print {new_id new_name \
       new_extension editable_p} {

    set db [ns_db gethandle subquery]

    set return_string ""

    set selection [ns_db select $db "select
    format_id, format_name, format_extension
    from asp_approved_formats"]

    set format_count 0

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query
        incr format_count

        append return_string "
        Convert from $format_name to $new_name:<BR>
        <INPUT TYPE=TEXT SIZE=100 NAME=$format_id.$new_id> <P>

        Convert from $new_name to $format_name:<BR>
        <INPUT TYPE=TEXT SIZE=100 NAME=$new_id.$format_id> <P>
        "
    }

    if {[string compare $editable_p t] == 0} {
        append return_string "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    }

    ns_db releasehandle $db

    return $return_string
}
```

B.5 Mechanisms

```
proc archive_mechanism_clause_print {clause_id clause_num} {

    set db [ns_db gethandle subquery]
```

```
    append return_string "
    <INPUT TYPE=HIDDEN NAME=clause_id_$clause_num VALUE=$clause_id>

    <SELECT NAME=n_people$clause_num>\n
    "

    for {set i 0} {$i <= 10} {incr i} {
        append return_string "<OPTION VALUE=$i> $i \n"
    }

    append return_string "
    </SELECT>

      people from the realm  

    <SELECT NAME=realm_id$clause_num>
    <OPTION VALUE=\"\"> Select a realm
    "

    #
    # Get the group from the group_id
    #

    set selection [ns_db select $db "select realm_id, realm_name
    from asp_realms
    order by realm_name"]

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query

        append return_string "<OPTION VALUE=$realm_id> $realm_name\n"
    }

    append return_string "</SELECT><BR>\n"

    ns_db releasehandle $db

    return $return_string
}

proc archive_mechanism_clause_request {clause_id mechanism_id \
       mechanism_clause_n_people mechanism_clause_group} {
    # Inserts the mechanism request into the database.

    set db [ns_db gethandle subquery]

    set mechanism_id_count [database_to_tcl_string $db "select count(*)
```

```
    from asp_mechanisms where mechanism_id = $mechanism_id"]

    if {$mechanism_id_count == 0} {

        # Can't find the mechanism!

        ns_db dml $db "insert into asp_mechanisms
        (mechanism_id, mechanism_name)
        values
        ($mechanism_id, 'mechanisms don''t have names yet!')"
    }

    ns_db dml $db "insert into asp_mechanism_clauses
    (mechanism_clause_id, mechanism_id,
       mechanism_clause_n_people, mechanism_clause_group)
    values
    ($clause_id, $mechanism_id,
       $mechanism_clause_n_people, $mechanism_clause_group)"

    ns_db releasehandle $db
}
```

B.6 Metadata

```
proc archive_metadata_print {new_file_identifier file_extension} {

    return "

    <H3>Article information</H3>

    <INPUT TYPE=HIDDEN NAME=file_extension VALUE=\"$file_extension\">

    <INPUT TYPE=HIDDEN NAME=article_id VALUE=\"$new_file_identifier\">

    Date on which article was created initially:<BR>
    <INPUT TYPE=TEXT SIZE=10 NAME=create_date VALUE=\"1999-01-31\"><P>

    Version:<BR>
    <INPUT TYPE=TEXT SIZE=10 NAME=version VALUE=\"1.0\"><P>

    Full title of the article: <BR>
    <INPUT TYPE=TEXT SIZE=100 NAME=title><P>

    Authors' and contributors' names: <BR>
    <INPUT TYPE=TEXT SIZE=100 NAME=authors><P>
```

```
    Keyword description: <BR>
    <INPUT TYPE=TEXT SIZE=100 NAME=keywords><P>

    Searchable description of the article (The abstract would be great):<BR>
    <TEXTAREA ROWS=10 COLS=95 NAME=description>
    </TEXTAREA><P>

    "
}
```

B.7 Realms

```
proc archive_realm_request {db \
   realm_type realm_name realm_parent_id requester_id} {

    set approval_id [database_to_tcl_string $db "select
       asp_approval_sequence.nextval from dual"]

    set realm_id [database_to_tcl_string $db "select
       asp_realm_sequence.nextval from dual"]

    set mechanism_id [database_to_tcl_string $db "select mechanism_id
    from asp_mechanism_map
    where table_name = 'asp_realms'
    and column_name = 'realm_parent'
    and column_value = '$realm_parent_id'"]

    # If there's no mechanism_id,
    # then it requires a system person as a default.

    if {![info exists mechanism_id] || $mechanism_id == ""} {
        set mechanism_id 5
    }

    ns_db dml $db "insert into asp_approvals
    (approval_id, approval_mechanism_id)
    values
    ($approval_id, $mechanism_id)"

    ns_db dml $db "insert into asp_realms
    (realm_id, realm_type, realm_name, realm_parent,
       realm_requester, realm_approval)
    values
    ($realm_id, '$realm_type', '$realm_name', $realm_parent_id,
        $requester_id, $approval_id)"
```

```
    # Mechanism id right now isn't implemented.
    # Defaults to system mechanism.

    set mechanism_id 5

    ns_db dml $db "insert into asp_mechanism_map
    (table_name, column_name, column_value, mechanism_id)
    values
    ('asp_realms', 'realm_parent', '$realm_id', $mechanism_id)"

}

proc archive_realm_print {realm_id editable_p} {
    # Prints the HTML for the archive realm.

    set return_string ""
    set db [ns_db gethandle subquery]

    append return_string "
    <INPUT TYPE=HIDDEN NAME=realm_id VALUE=$realm_id>
    <P>
    Realm name:<BR>
    <INPUT TYPE=TEXT SIZE=50 NAME=realm_name>
    <P>
    Realm parent:<BR>
    <SELECT NAME=realm_parent>
    "

    set selection [ns_db select $db "select realm_name, realm_id
    from asp_realms
    order by upper(realm_name)"]

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query

        append return_string "<OPTION VALUE=$realm_id> $realm_name \n"
    }

    append return_string "</SELECT>\n"

    if {[string compare $editable_p t] == 0} {
        append return_string "<P>
        <INPUT TYPE=SUBMIT VALUE=Submit>\n"
    }

    ns_db releasehandle $db
```

```
    return $return_string
}

proc archive_realm_membership_request {user_id realm_id} {

    set db [ns_db gethandle subquery]

    if {[catch {set mechanism_id [database_to_tcl_string $db "select
    mechanism_id
    from asp_mechanism_map
    where table_name = 'asp_realms'
    and column_name = 'realm_id'
    and column_value = $realm_id"]} error]} {

        set mechanism_id 5
    }

    set approval_id [database_to_tcl_string $db "select
       asp_approval_sequence.nextval from dual"]

    ns_db dml $db "insert into asp_approvals
    (approval_id, approval_mechanism_id)
    values
    ($approval_id, $mechanism_id)"

    ns_db dml $db "insert into asp_realm_user_map
    (realm_id, user_id, approval_id)
    values
    ($realm_id, $user_id, $approval_id)"

    ns_db releasehandle $db
}

proc archive_realm_membership_print {editable_p} {
    set return_string ""

    set db [ns_db gethandle subquery]

    set selection [ns_db select $db "select realm_name, realm_id
    from asp_approved_realms
    order by upper(realm_name)"]

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query

        if {[string first @ $realm_name] == -1} {
            append return_string "<INPUT TYPE=CHECKBOX NAME=realms
```

```
                VALUE=$realm_id> $realm_name <BR>\n"
        }
    }

    if {[string compare $editable_p t] == 0} {
        append return_string "<P>
        <INPUT TYPE=SUBMIT NAME=Submit>\n"
    }

    return $return_string
}
```

B.8 System definitions

```
proc required {} {
    return "\[ * \]"
}

proc archive_header {header_string} {
    return "[ad_header "$header_string"]
    <H2>$header_string</H2>
    Part of [archive_system_name]
    <HR>
    "
}

proc archive_footer {} {
    return "[ad_footer]"
}

proc archive_system_name {} {
    return "[ad_system_name]"
}

proc archive_system_url {} {
    return "[ad_url]"
}

proc archive_system_owner {} {
    return "[ad_system_owner]"
}

proc archive_system_path {} {
    return "/web/archive/www"
}

proc archive_system_pvt_path {} {
```

```
    return "[archive_system_path]/pvt"
}
```

Bibliography

[1] American Medical Association. Principles governing advertising in a.m.a. publications. *http://www.ama-assn.org/public/journals/ratecard/principl.htm*, 1998.

[2] Declan Butler. The writing is on the web for science journals in print. *Nature, http://www.nature.com/server-java/Propub/nature/397195A0.frameset?context=toc*, 1999.

[3] Eric F. Celeste. Discussions with eric f. celeste of the mit libraries. *Discussion through email.*, 1999.

[4] Online Computer Library Center. Permanent uniform resource locators. *http://purl.org*, 1999.

[5] World Wide Web Consortium. Resource description framework. *http://www.w3c.org/rdf*, 1999.

[6] Michael Bryzek et. al. The virtual compassion corps. *http://CompassionCorps.org*, 1999.

[7] Philip Greenspun. The book behind the book behind the book... *http://photo.net/wtr/dead-trees/story.html*, 1997.

[8] The Dublin Core Metadata Initiative. The dublin core metadata initiative homepage. *http://www.purl.org/dc*, 1999.

[9] Richard K. Johnson. Sparc whitepaper. *http://www.arl.org/sparc/whitepaper.html*, 1999.

[10] Los Alamos National Laboratories. e-print archive. *http://xxx.lanl.gov*, 1995.

[11] Lawrence Lessig. The code is the law. *The Industry Standard*, 1999.

[12] Stanford University Libraries and Academic Information Resources. High wire press. *http://www.highwirepress.com*, 1999.

[13] ArsDigita LLC. The arsdigita community system. *http://software.arsdigita.com*, 1999.

[14] Arnold P. Lutzker. What the digital millennium copyright act and the copyright term extension act mean for the library community. *http://www.arl.org/info/frn/copy/primer.html*, 1999.

[15] Ann Okerson. Back to academia? the case for american universities to publish their own research. *LOGOS*, 1991.

[16] Ann Okerson. With feathers: Effects of ownership on scholarly publishing. *College & Research Libraries*, 1991.

[17] America OnLine. *AOLserver*. America OnLine, http://www.aolserver.com, 1995.

[18] MIT Registrar. Web student information system. *http://student.mit.edu*, 1999.

[19] Mauricio
Roman. Noble tenure case will go to trial. *http://tech.mit.edu/V110/N12/noble.12n.html*, 1990.

[20] Gerald Jay Sussman. A computational model of skill acquisition. *MIT Libraries*, 1973.

[21] Ann Wolpert. Conversation about obsolescence of library storage mechanisms. *Laboratory for Computer Science 35th Anniversary*, 1999.

[22] XML.com. Xml news feed: All you can parse. *http://www.xml.com*, 1999.

[23] Yahoo! Yahoo! homepage. *http://www.yahoo.com*, 1999.